PCT ORGANISATION MONDIALE DE LA PROPRIETE INTELLECTUELLE
Bureau international

B1

DEMANDE INTERNATIONALE PUBLIEE EN VERTU DU TRAITE DE COOPERATION EN MATIERE DE BREVETS (PCT)

(51) Classification internationale des brevets 7 : G06F 9/46, 17/50

A1

(11) Numéro de publication internationale: WO 00/60460

(43) Date de publication internationale: 12 octobre 2000 (12.10.00)

(21) Numéro de la demande internationale: PCT/FR00/00824

(22) Date de dépôt international: 31 mars 2000 (31.03.00)

(30) Données relatives à la priorité:
99/04182 2 avril 1999 (02.04.99) FR

(71) Déposant *(pour tous les Etats désignés sauf US)*: THOMSON-CSF [FR/FR]; 173, boulevard Haussmann, F-75008 Paris (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(US seulement)*: MATTIOLI, Juliette [FR/FR]; Thomson-CSF Propriété Intellectuelle, Dépt. Brevets, 13, avenue du Prés. Salvador Allende, F-94117 Arcueil Cedex (FR). GUETTIER, Christophe [FR/FR]; Thomson-CSF Propriété Intellectuelle, Dépt. Brevets, 13, avenue du Prés. Salvador Allende, F-94117 Arcueil Cedex (FR). JOURDAN, Jean [FR/FR]; Thomson-CSF Propriété Intellectuelle, Dépt. Brevets, 13, avenue du Prés. Salvador Allende, F-94117 Arcueil Cedex (FR).

(74) Mandataire: CHAVERNEFF, Vladimir; Thomson-CSF Propriété Intellectuelle, Dépt. Brevets, 13, av. du Prés. Salvador Allende, F-94117 Arcueil Cedex (FR).

(81) Etats désignés: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, brevet ARIPO (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Publiée

*Avec rapport de recherche internationale.*

*Avant l'expiration du délai prévu pour la modification des revendications, sera republiée si des modifications sont reçues.*

(54) Title: GENERIC AID METHOD FOR PLACING SIGNAL PROCESSING APPLICATIONS ON PARALLEL COMPUTERS

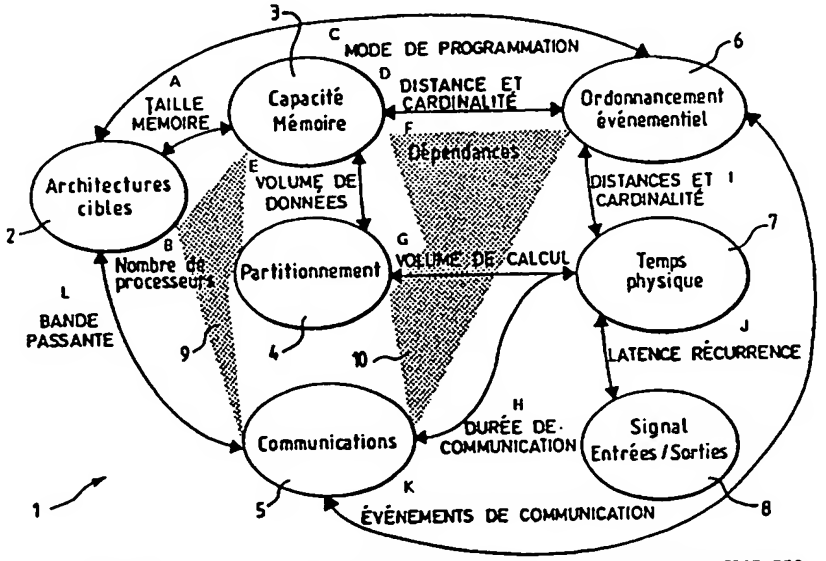
(54) Titre: PROCEDE GENERIQUE D'AIDE AU PLACEMENT D'APPLICATIONS DE TRAITEMENT DE SIGNAL SUR CALCULATEURS PARALLELES

(57) Abstract

The inventive method for systematic signal processing on a homogenous parallel architecture multiprocessor computer consists in establishing a model for each physical and functional constituent of the application, whereby said model is defined by a set of relations on different variables relating to said constituent, in order to model the constraints and resolve said relations in a concurrent manner and deduce at least one solution therefrom.

(57) Abrégé

Le procédé de l'invention, destiné au traitement de signal systématique sur un calculateur multiprocesseur à architecture parallèle homogène, consiste à établir, pour chaque constituant fonctionnel et physique de l'application, un modèle défini par un ensemble de relations sur les différentes variables relatives à ce constituant, afin de modéliser les contraintes, puis à résoudre de façon concurrente ces relations, et à en déduire au moins une solution.




A_MEMORY SIZE
B_NUMBER OF PROCESSORS
C_PROGRAM MODE
D_DISTANCE AND CARDINALITY
E_VOLUME OF DATA
F_DEPENDENCES
G_VOLUME OF CALCULATION
H_LENGTH OF COMMUNICATION
I_DISTANCE AND CARDINALITY
J_RECURRENCE LATENCY
K_COMMUNICATION EVENTS
L_BANDWIDTH
2_TARGET ARCHITECTURES
3_STORAGE CAPACITY
4_PARTITIONING
5_COMMUNICATIONS
6_EVENT ORDERING
7_PHYSICAL TIME
8_INPUT/OUTPUT SIGNAL

*UNIQUEMENT A TITRE D'INFORMATION*

Codes utilisés pour identifier les Etats parties au PCT, sur les pages de couverture des brochures publiant des demandes internationales en vertu du PCT.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| AL | Albanie | ES | Espagne | LS | Lesotho | SI | Slovénie |
| AM | Arménie | FI | Finlande | LT | Lituanie | SK | Slovaquie |
| AT | Autriche | FR | France | LU | Luxembourg | SN | Sénégal |
| AU | Australie | GA | Gabon | LV | Lettonie | SZ | Swaziland |
| AZ | Azerbaïdjan | GB | Royaume-Uni | MC | Monaco | TD | Tchad |
| BA | Bosnie-Herzégovine | GE | Géorgie | MD | République de Moldova | TG | Togo |
| BB | Barbade | GH | Ghana | MG | Madagascar | TJ | Tadjikistan |
| BE | Belgique | GN | Guinée | MK | Ex-République yougoslave de Macédoine | TM | Turkménistan |
| BF | Burkina Faso | GR | Grèce | | | TR | Turquie |
| BG | Bulgarie | HU | Hongrie | ML | Mali | TT | Trinité-et-Tobago |
| BJ | Bénin | IE | Irlande | MN | Mongolie | UA | Ukraine |
| BR | Brésil | IL | Israël | MR | Mauritanie | UG | Ouganda |
| BY | Bélarus | IS | Islande | MW | Malawi | US | Etats-Unis d'Amérique |
| CA | Canada | IT | Italie | MX | Mexique | UZ | Ouzbékistan |
| CF | République centrafricaine | JP | Japon | NE | Niger | VN | Viet Nam |
| CG | Congo | KE | Kenya | NL | Pays-Bas | YU | Yougoslavie |
| CH | Suisse | KG | Kirghizistan | NO | Norvège | ZW | Zimbabwe |
| CI | Côte d'Ivoire | KP | République populaire démocratique de Corée | NZ | Nouvelle-Zélande | | |
| CM | Cameroun | | | PL | Pologne | | |
| CN | Chine | KR | République de Corée | PT | Portugal | | |
| CU | Cuba | KZ | Kazakstan | RO | Roumanie | | |
| CZ | République tchèque | LC | Sainte-Lucie | RU | Fédération de Russie | | |
| DE | Allemagne | LI | Liechtenstein | SD | Soudan | | |
| DK | Danemark | LK | Sri Lanka | SE | Suède | | |
| EE | Estonie | LR | Libéria | SG | Singapour | | |

# PROCEDE GENERIQUE D'AIDE AU PLACEMENT D'APPLICATIONS DE TRAITEMENT DE SIGNAL SUR CALCULATEURS PARALLELES

La présente invention se rapporte à un procédé générique d'aide au placement d'applications de traitement de signal sur calculateurs parallèles.

Le placement (« Mapping » en anglais) consiste à distribuer les données et les traitements liés à un traitement, tel qu'une application de traitement de signal, sur un calculateur, en général un calculateur à architecture parallèle. Ce placement est statique, car l'ensemble des choix du placement est pris avant l'exécution de l'application placée, contrairement au placement dynamique.

On connaît, pour réaliser le placement, de nombreux outils et environnements de programmation. On citera, entre autres, l'outil SynDEx de l'INRIA pour le traitement de signal et d'image, l'outil PTOLEMY de l'Université de Berkeley, le HPF pour le calcul scientifique, le « FX compiler », le GEDEA de la Société LOCKHEED MARTIN, ... Cependant, peu d'outils connus permettent l'automatisation complète du placement. De plus, même si l'objectif général des outils tels que SynDEx ou GEDEA est de fournir à l'utilisateur une aide au développement et à l'optimisation d'applications temps réel implémentées sur une architecture multiprocesseurs, généralement hétérogène, en vue de la réalisation rapide de prototypes, ces outils ne gèrent qu'un niveau de granularité grossier (la granularité étant le degré de finesse et de précision que l'on veut obtenir pour une application et son implémentation sur une architecture donnée). L'environnement Ptolemy est essentiellement un environnement de simulation et de prototypage de systèmes hétérogènes intégrant du matériau et du logiciel.

De plus, tous ces systèmes connus permettent, en général, d'estimer les performances d'un placement pour une solution donnée, tout en indiquant le réseau de communication entre processeurs le plus efficace, ainsi qu'un code généré automatiquement pour chacun des processeurs du système.

C'est pourquoi, les langages dédiés, tel le HPF, proposent des primitives de placement manuel. A partir de ces primitives, le programmeur doit déterminer lui-même le bon placement. Il en résulte que l'utilisation des

ressources offertes par des super-calculateurs comportant de très nombreux processeurs est loin d'être optimale, tant la fonction de placement est complexe. En effet, cette fonction inclut le découpage, la distribution et l'alignement des données sur les différents processeurs, la répartition des tâches de calcul et de communication, ainsi que leur ordonnancement dans le temps. De plus, chaque choix relatif à l'une de ces fonctions est intimement lié à l'architecture du calculateur et aux caractéristiques physiques des architectures parallèles.

La présente invention a pour objet un procédé générique d'aide au placement d'applications de traitement de signal systématique sur un calculateur à architecture parallèle homogène, procédé qui permette d'obtenir automatiquement au moins une solution optimisée de placement, à un niveau de granularité aussi fin que possible, et ce, à partir d'une description fonctionnelle complète de l'application, et du calculateur utilisé.

Le procédé conforme à l'invention consiste, pour chaque constituant fonctionnel et physique de l'application, à établir un modèle défini par un ensemble de relations sur les différentes variables relatives à ce constituant, afin de modéliser les contraintes, à résoudre de façon concurrente les relations ainsi établies, à en déduire au moins une solution, et, en cas d'obtention de plusieurs solutions, à choisir celle optimisant au moins un critère.

Les contraintes sont celles relatives aux sous-fonctions de la fonction de placement, à savoir : le partitionnement, l'alignement, la distribution de données et le séquencement des traitements.

La présente invention sera mieux comprise à la lecture de la description détaillée d'un mode de mise en œuvre, pris à titre d'exemple non limitatif et illustré par le dessin annexé, dont la figure unique est un schéma fonctionnel de la fonction placement, mise en œuvre conformément à l'invention.

La présente invention se rapporte au traitement de signal systématique, c'est-à-dire non conditionnel, non soumis à des ordres ou actions extérieurs. Ce traitement est, en outre, déterministe et structuré. Ce traitement peut être, par exemple, de la compression d'impulsions ou le calcul de transformées de Fourier (FFT).

Les applications de traitement du signal systématique sont formées de séquences de tâches, que l'on peut exprimer par des nids de boucles (boucles imbriquées et à bornes définies) bien structurés et parallèles. Chaque nid de boucles contient un appel à une procédure ou macro-instruction correspondant en général à une transformation de tableau c'est-à-dire à une fonction d'une librairie de traitement du signal telle qu'une FFT. Une telle transformation a été décrite dans le brevet français n° 2 732 787. Les traitements sont réguliers (non soumis à des tests extérieurs) et s'effectuent sur des signaux multi-dimensionnels, les données sont organisées en grands tableaux dont les dimensions (par exemple source, fréquence, temps de récurrence, temps de pointage) portent les vecteurs sur lesquels vont s'effectuer les traitements individuels. Le tableau s'adapte facilement aux dimensions du système de capteurs, et permet de donner informatiquement la formulation mathématique des traitements. Ainsi, les indices des variables composant les formules deviennent des indices de tableaux.

Ces procédures ont un coût d'exécution fixé, inclus dans la spécification de l'application. Celle-ci est globalement représentée par un graphe de flot de données acyclique. L'application est, en effet, sous forme d'assignation unique, c'est-à-dire que chaque élément de tableau n'est mis qu'une seule fois à jour par l'application. Dans une mise en œuvre parallèle, la distribution de ce grand tableau sur les nœuds de calcul change d'un traitement au suivant, provoquant ainsi un problème classique du parallélisme : le changement d'axe, ou « corner turn », consommant beaucoup de ressources de communication.

Pour pouvoir mettre en œuvre l'invention, il faut décrire fonctionnellement l'application de traitement de signal, et plus particulièrement les composantes de cette applications, à savoir les tâches.

Une tâche, aussi appelée, routine, procédure ou traitement, accepte en entrée et en sortie un ou plusieurs flux de données. Ces flux sont définis à partir des tableaux d'entrée et de sortie, de la façon exposée dans le susdit brevet français n° 2 732 787, et que l'on rappelle succinctement ci-dessous. Sur chaque tableau considéré, un flux représente les données accédées en lecture ou écriture par un et un seul traitement élémentaire. L'ensemble de ces données constitue un accès élémentaire ou domaine de

la transformation élémentaire. Un même traitement est réitéré sur un espace d'itération défini par l'application. Certaines propriétés sont associées à ce nœud du traitement :

- **La formulation du traitement** : elle décrit la formule de la transformation élémentaire. Les données de sortie sont exprimées en fonction des données d'entrée. Les accès élémentaires en lecture ou en écriture sont spécifiés en fonction des indices des tableaux. Les dimensions des tableaux et l'espace mémoire requis pour exécuter un traitement sur tout l'espace d'itération du traitement sont également spécifiés.
- **La complexité du calcul** : elle exprime la puissance de calcul requise par le nœud de traitement en opérations/seconde. Elle est liée à l'architecture utilisée, et représente une donnée d'entrée de l'application.

Le flux de données peut être conditionné par les données ou les indices des tableaux (il dépend de l'application). Comme pour les nœuds, des propriétés sont associées aux flux de données :

- **Le codage des données**. Celles-ci sont codées sur un certain nombre de bits (12-16-32-64-...), elles représentent généralement des nombres réels à virgule fixe ou bien des nombres complexes (codés sur deux nombres entiers).
- **La récurrence d'un flux**. Cette valeur est une fonction de la puissance de calcul des traitement dont le flux provient des récurrences des flux entrants de ce même traitement.
- L'acquisition est considérée comme une tâche à part entière comprenant un flux de sortie, et une récurrence (fréquence d'acquisition).
- **Le nombre de données liées à une transformation élémentaire**. Il définit le nombre de données que requiert la transformation élémentaire du nœud destination. Ces données sont dites produites par le nœud source et dites consommées par le nœud destination. Toutefois, il arrive que des données soient calculées et non utilisées par la suite (on n'exploite pas la totalité des données d'une bibliothèque, qui peut être

d'usage général), ce qui n'implique pas de duplication de calcul.

Les accès élémentaires à un tableau sont des fonctions affines des indices de ce tableau, des constantes et des variables scalaires privées. L'espace d'itération d'un traitement est quant à lui complètement défini par des fonctions affines portant uniquement sur les indices des différents tableaux.

Plus précisément, une tâche se décompose en deux parties :

- **L'espace d'itérations externe** décrivant le domaine de calculs. L'une des dimensions peut être infinie. Elle représente le temps. Ce domaine est représenté par un nid de boucles parfaitement imbriqué et totalement parallèle. Il n'y a aucune dépendance en écriture, par contre, il peut exister des recouvrements en lecture. C'est ce domaine de calculs qu'il faut placer et ordonnancer sur la machine parallèle.
- **L'espace d'itérations interne** décrivant l'ensemble des données qui sont utiles au calcul de la macro-instruction ou procédure. Les fonctions d'accès aux éléments de tableau sont des fonctions pseudo-affines. Des fonctions modulo sont parfois utilisées pour prendre en compte le caractère cyclique des capteurs pouvant être reliés au calculateur.

On peut utiliser comme formalisme de description fonctionnelle de l'application de TSS (traitement de signal systématique) un graphe de flot de données (Data Flow Graph) qui peut provenir de n'importe quel formalisme classique de description d'application du signal sous réserve qu'il contienne les informations précisées ci-dessus. En particulier, on peut décrire l'application à partir des langages suivants :

- Le langage « Array-Ol » (exposé dans le susdit brevet français 2 732 787),
- Le langage ALPHA,
- Un langage décrivant un ensemble de nids de boucles parfaitement imbriqués,
- Ou le formalisme de description de type MD/SDF.

Le placement consiste à distribuer automatiquement les opérations de traitement de signal à effectuer sur un flot de données, et ces

données elles-mêmes, sur un calculateur à architecture multiprocesseurs parallèle en tenant compte des différentes contraintes de ressources matérielles ainsi que des performances imposées au calculateur.

L'architecture parallèle dont il est question ici est une architecture parallèle homogène, dans laquelle tous les processeurs sont identiques, de type SIMD/SPMD (« Single Instruction/Program Multiple Data »), c'est-à-dire dans laquelle tous les processeurs exécutent la même instruction ou la même séquence d'instructions (par exemple un programme) sur des données différentes. Le routage de l'information entre les différents processeurs est de type statique, c'est-à-dire que les chemins de données entre processeurs sont imposés avant l'initialisation de chaque mode (ils sont définis lors de la compilation de l'application). A un instant donné, les macro-instructions exécutées en parallèle sur chacun des processeurs sont identiques. Les données nécessaires au traitement de la macro-instruction doivent résider dans la mémoire locale du processeur qui l'exécute.

Pour les applications de traitement de signal, les « dimensions » de l'architecture du calculateur utilisé sont des contraintes impératives du placement. Cependant, ces contraintes ne sont pas prises en compte par les procédés classiques de placement automatique (tels que les procédés cités ci-dessus) et ne sont donc pas traitées à cet effet dans l'état de l'art. Les paramètres caractéristiques desdites dimensions sont :

- Le nombre de processeurs disponibles, qui sont tous de puissance égale.
- La puissance d'un processeur. Pour les applications de traitement de signal en temps réel, le temps de latence des calculs (temps au bout duquel les résultats de ces calculs sont disponibles) est très important. Ce temps peut être borné par une valeur maximale, et il dépend de la puissance du processeur qui réalise le calcul. Cette puissance est exprimée en nombre de cycles de calcul par seconde.
- La mémoire disponible. Elle est répartie uniformément sur l'ensemble des processeurs du calculateur.
- Les caractéristiques de communication entre processeurs. Elles sont définies en termes de bande passante, c'est-à-dire en nombre de cycles machine nécessaires à assurer la

communication d'un paquet de données entre les processeurs d'une paire de processeurs. La durée des communications n'est alors pas fonction de la topologie du calculateur.

Le placement dont il est question ici comprend les quatre sous-fonctions de partitionnement d'alignement, de distribution et de séquencement. Jusqu'à présent, ces quatre sous-fonctions étaient traitées chacune séparément. Par contre, la présente invention prévoit de traiter simultanément ces sous-fonctions et de façon concurrente. Ce placement permet de trouver l'adéquation entre un programme (dont le parallélisme est spécifié ou non) et un calculateur à architecture parallèle homogène telle que spécifiée ci-dessus. Il consiste à distribuer les traitements et les données sur les différents processeurs du calculateur et à établir leur séquencement, en optimisant le parallélisme de l'application.

Ce placement comprend la détermination des diverses contraintes liées à l'application. Ces contraintes sont, d'une part, des contraintes « applicatives » (liées à la taille des tâches élémentaires spécifiques du traitement de signal systématique), d'autre part des contraintes liées à l'architecture du calculateur (nombre de processeurs, capacité mémoire, topologie du réseau de processeurs et débit de données, et enfin des contraintes liées à l'exécution (ordonnancement affine, recouvrement entre les communications de données et les calculs effectués).

La modélisation par contraintes consiste essentiellement à établir, pour chaque contrainte, une relation entre au moins deux variables ou une relation entre une variable et une valeur donnée (généralement un seuil). Cette relation est une relation linéaire (généralement un polynôme du 1$^{er}$ degré).

Le procédé de l'invention, partant de cette modélisation, effectue la résolution concurrente (non séquentielle) de tous les modèles, pour en déduire une ou plusieurs solutions satisfaisant toutes les contraintes. Dans le cas de plusieurs solutions satisfaisantes, on peut choisir celle satisfaisant de la meilleure façon un ou plusieurs critères (dont des exemples sont cités ci-dessous), et on peut avantageusement procéder de façon heuristique.

Les langages de spécifications d'applications de signal, les polynômes en nombres entiers et les applications affines permettent de modéliser précisément les fonctionnalités du placement. L'algèbre linéaire

permet de construire les différents modèles au niveau de granularité requis par la complexité du problème général. Ces modèles sont issus de l'état de l'art du placement automatique [J. Li & M. Chen, « The data alignment phase in compiling programs for distributed memory machines », Journal of Parallel and Distributed Computing, p. 213-221, Vol. 13, 1991 ; P. Feautrier, « Toward Automatic Distribution », Parallel Processing Letters, p. 233-244, Vol. 4, n° 3, 1994 ; M. Dion, « Alignement et distribution en parallélisation automatique », Thèse Informatique, ENS. LYON, 1996] et ont fait l'objet d'une adaptation au contexte applicatif du signal. Ils sont exprimés à l'aide de contraintes linéaires et non linéaires intégrables au procédé de l'invention. Ce procédé s'impose alors d'une part par sa capacité à traiter l'algèbre propre aux différents modèles et d'autre part par sa dimension de langage qui facilite l'expression et la composition d'heuristiques et de contraintes spécifiques à un domaine, et/ou de stratégies complexes permettant le contrôle des étapes de résolution. Les données manipulées par le système de placement sont donc des éléments de tableaux, des macro-instructions, dont il faut nécessairement modéliser le comportement selon les fonctions à résoudre. L'ensemble des modèles décrits ci-dessous sont entièrement formalisés à partir de l'algèbre linéaire, permettant le contrôle de la granularité pour chaque modèle. Chaque modèle définit une composante de la compilation d'une application de traitement du signal sur machine parallèle.

- La distribution des calculs et des données sur les processeurs, est en fait un problème de ressources disjonctives (exclusives les unes des autres).
- L'ordonnancement des traitements en tenant compte des ressources mémoire et communications, représentées par des contraintes capacitives.

Le but est d'optimiser différents critères comme la latence de l'application ou le coût (financier) de l'architecture cible. De plus, de nombreux modèles spécifiques au problème de placement ont été développés et viennent compléter la description du problème, comme par exemple les communications ou le temps physique. Compte tenu du nombre de tâches et du nombre de données à considérer lors du placement, chaque modèle est défini en intention plutôt qu'en extension. La possibilité de

travailler à plusieurs niveaux de granularité s'avère fondamentale pour le problème de placement, et l'on utilise pour cela une formulation algébrique du partitionnement. Celui-ci fixe la granularité des autres modèles, il entretient donc de nombreuses relations au sein même du modèle conceptuel. De plus, les contraintes de dépendances relient à de nombreuses reprises plusieurs modèles, ce sont donc des contraintes globales et la clef de voûte du problème à résoudre. Enfin, on dispose d'heuristiques locales à un ou plusieurs modèles. Cependant, on ne connaît pas d'heuristique globale. Le cœur de la présente invention utilise l'approche multi-modèles par contraintes concurrentes [J. Jourdan, F. Fages, D. Rozzonelli & A. Demeure, « Data Alignment and Task Scheduling On Parallel Machines Using Concurrent Model-based Programming », Proc. ILPS 94, 1994], ce qui permet d'appréhender le problème du placement automatique de manière globale.

Selon le procédé de l'invention, les modèles sont établis à raison d'un modèle par constituant, qu'il soit fonctionnel ou physique. Par définition, un modèle doit être vu comme l'ensemble des spécifications du comportement du constituant qu'il modélise.

On a représenté sur le schéma fonctionnel de la figure unique du dessin les différents modèles mis en œuvre pour la fonction « placement » référencée 1 dans son ensemble. Ces modèles sont : l'architecture des processeurs cibles (2), la capacité de la mémoire (3), le partitionnement des flots de données (4), les communications inter-processeurs (5), l'ordonnancement événementiel ou séquencement des calculs (6), le temps physique ou temps de calcul (7) et les entrées-sorties de signaux (8).

Les différents liens établis entre ces modèles sont de deux sortes : les « hyperliens » représentés par des flèches complexes (9, 10) en forme de polygones irréguliers , qui relient chacune plusieurs modèles ensemble, et des liens simples, représentés par des traits fléchés chacun à leurs extrémités et reliant chacun deux modèles.

La flèche complexe 9, qui correspond au critère « nombre de processeurs », relie les modèles 2, 3, 4 et 5. La flèche complexe 10, qui correspond au critère « dépendances », relie les modèles 3, 4, 5 et 6.

Le modèle 2 est relié par des liens simples aux modèles (3) (critère « taille mémoire »), 5 (critère « bande passante ») et 6 (critère « mode de programmation »).

Le modèle 3 et relié par des liens simples aux modèles 4 (critère « volume de données ») et 6 (critère « distance et cardinalité).

Le modèle 4 est relié par un lien simple au modèle 7 (critère « volume de calcul »).

Le modèle 5 est relié par des liens simples aux modèles 6 (critère « événements de communication ») et 7 (critère « durée de communication »).

Le modèle 6 est relié par un lien simple au modèle 7 (critère « distance et cardinalité »).

Enfin, le modèle 7 et relié par un lien simple au modèle 8 (critère « latence et récurrence »).

Dans les modèles décrits ci-dessus, les spécifications du comportement des différents constituants de ces modèles, sont exprimés à partir de relations mathématiques. On peut donc en déduire que les modèles sont identifiés à l'ensemble des relations définies sur leurs variables. Ces relations sont soit des primitives du langage utilisé (primitives faisant partie d'une bibliothèque de relations), soit des relations définies par l'utilisateur.

Du fait que les modèles constituent eux-mêmes l'essentiel du procédé, les propriétés du paradigme relationnel (ensemble des règles régissant les relations pouvant être établies entre les modèles) ont des conséquences immédiates sur les propriétés des composantes fonctionnelles du procédé. Les propriétés du paradigme relationnel sont les suivantes :

- **Description formelle** : Les relations représentent une description formelle du comportement du constituant. En effet, pour chacune des sous-fonctions du placement, une modélisation mathématique a été spécifiée formellement. Dans la plupart des cas, elle est issue des travaux des spécialistes des parallélisations, et elle a été adaptée non seulement au cadre applicatif qu'est le traitement du signal mais étendue au contexte de la modélisation concurrente.

- « **Adirectionalité** » : Le concept de relation permet d'abandonner le paradigme fonctionnel basé sur la distinction des entrées/sorties. Une relation assure une réversibilité totale des arguments. Ceci permet de ne faire la distinction qu'à l'exécution, en fonction de la nature des arguments (connus et inconnus).
- « **Compositionalité** » : La composition des modèles relationnels est tout simplement la conjonction logique des relations qui constituent le modèle. Ceci implique une sémantique simple de la compositionalité. L'ensemble des solutions d'un modèle composite est tout simplement l'intersection des solutions des modèles.

Elles contribuent à l'universalité du programme. Les propriétés du procédé induites sont alors :

- **Un domaine d'utilisation élargi** : Un modèle peut être utilisé dans plusieurs contextes en fonction du but à réaliser.
- **Une interchangeabilité accrue** : Un modèle peut être modifié ou complètement redéfini par la donnée d'une nouvelle spécification sans avoir à intervenir sur les autres modèles.
- **Une compositionalité intrinsèque** : Le modèle d'un système est construit à partir du modèle de ses composants.
- **Une maintenabilité simple** : La maintenabilité reste locale à chaque modèle.
- **Une extensibilité aisée** : Etendre un système revient à le composer avec le système existant.

Dans toutes les solutions logicielles connues de parallélisation, l'état d'un système est caractérisé par le contenu de la mémoire à un instant donné. Les opérations élémentaires sont la lecture et l'écriture sur, ou à partir de la mémoire. L'état d'un système n'est alors caractérisé que par l'ensemble des valeurs des cases mémoires associées aux variables qui le composent. La différence fondamentale entre le procédé de l'invention et les autres solutions logicielles est la représentation de cette mémoire. Dans le cas de l'invention, la mémoire n'est pas réduite à un ensemble de cases mémoire mais constitue en elle-même une contrainte. Cette dernière est capable de fournir de l'information partielle sur l'ensemble des variables qui

composent le système. Il est intéressant de noter que tout le raisonnement mis en œuvre par les contraintes est basé sur ce paradigme de manipulation d'informations partielles. L'avantage des contraintes est tout simplement dû au fait que le système en cours d'élaboration peut prendre des décisions sans avoir à attendre qu'il soit entièrement déterminé.

La résolution d'applications industrielles ne se cantonne pas à une problématique bien définie, mais intègre la combinaison de plusieurs sous-problèmes. Il faut résoudre des problèmes d'optimisation combinatoire sur des problèmes multi-composants, multifonctions dans lesquels les contraintes sont très hétérogènes et où il faut considérer les différents éléments à différents niveaux de granularité. L'invention offre des solutions permettant la coexistence de modèles se recouvrant partiellement, se coordonnant et se décomposant.

Une alternative pour les problèmes multi-composants, multifonctions : la spécification de modèles pour chaque composant et pour chaque fonction qui se combinent ensuite lors de la résolution d'un but, permet d'apporter une alternative à la résolution de problèmes de systèmes. De plus, souvent les modèles sont très hétérogènes. Certains peuvent s'exprimer exclusivement à l'aide de contraintes linéaires, d'autres peuvent nécessiter des contraintes symboliques ou booléennes.

L'invention permet indépendamment de l'hétérogénéité des contraintes, par de simples interactions locales, de garantir une coordination globale du système.

Dans le cas des problèmes d'optimisation combinatoire, le procédé de l'invention offre une bonne solution technologique, car elle permet, lors de la résolution, l'utilisation concurrente de tous les modèles redondants.

En effet :

- La résolution de problèmes hautement combinatoires se résume rarement à une seule formalisation mathématique. Souvent, pour ne pas dire toujours, des formalisations complémentaires sont nécessaires. Un exemple pourrait être les formalisations redondantes qui tirent avantage des propriétés des solutions partielles, un autre exemple serait la prise en compte des symétries du problèmes, etc.

- Le problème qui se pose alors est d'utiliser en même temps toutes ces informations. Dans le contexte d'approches plus classiques, comme la recherche opérationnelle ou la programmation en nombres entiers, cette étape s'avère toujours très délicate. En effet, les programmes écrits dans un langage impératif nécessitent un temps de développement non négligeable et sont souvent difficiles à étendre et à modifier.

L'invention permet également de résoudre les problèmes relationnels des modèles à plusieurs niveaux de granularité.

L'efficacité de la réalisation finale dépend de trois paramètres. Tout d'abord, elle dépend crucialement de l'efficacité du système de contraintes utilisé, du contrôle que l'on a sur la recherche d'une solution (voir ci-dessous), mais aussi, considérablement, de la granularité considérée dans la modélisation. L'expérience montre qu'il est indispensable de considérer différents niveaux de granularité et de pouvoir raisonner sur les différents niveaux. Ici encore, l'invention offre une solution en permettant la description, la coexistence et la coordination de modèles à différents niveaux de granularité. Ce dernier point n'a pas encore été vraiment exploité dans des applications.

La fonction « placement » est alors modélisée au travers de différents modèles issus des travaux des spécialistes de la « parallélisation automatique » comme :

- Le **partitionnement** qui exprime la distribution des calculs et des données sur les processeurs.
- Les **dépendances** qui caractérisent les itérations accédant aux mêmes données.
- L'**ordonnancement** qui consiste à organiser l'exécution des traitements parallèles dans le temps.
- L'**interphase** qui donne les communications entre deux phases de calcul partitionnées.
- L'**architecture** qui est en fait un ensemble de paramètres comme le nombre de processeurs, la bande passante, ...
- Les **communications**.
- La **mémoire** qui définit au travers d'une contrainte de capacité pour laquelle on est sûr de pouvoir calculer une allocation.

- Le **signal temps réel** qui est constitué de la latence et des contraintes d'entrées/sorties (périodes, ...).

De manière générique, un modèle se compose de définitions de variables sur lesquelles reposent les contraintes propres aux modèles.

Le procédé de l'invention permet à la fois de résoudre le problème du placement automatique d'applications traitement de signal sur machines parallèles et permet à l'utilisateur de manipuler une solution fournie sans violation des contraintes posées par le système global. Cette approche s'inscrit dans un contexte de codesign et de prototypage virtuel. A partir d'une description formelle de l'application (comme par exemple un langage de type MD/SDF développé à Berkeley, une description fonctionnelle en ARRAY-OL ou une spécification à l'aide de nids de boucles) l'outil produira un pseudo-code générique de placement pour des machines cibles homogènes considérées. Ce pseudo-code sera ensuite directement interfaçable avec les différents compilateurs des architectures cibles.

Ce procédé peut donc permettre à l'utilisateur :

- **de saisir une solution partielle** : l'utilisateur saisit une solution partielle de placement et l'outil poursuit sa recherche en complétant la solution de placement qui sera validée par conception. Par exemple :
  - L'utilisateur a configuré sa machine avec un nombre insuffisant de processeurs pour le type de placement qu'il désire. Le procédé va permettre de trouver le nombre minimal de processeurs nécessaires au placement imposé dans l'ensemble des processeurs disponibles.
  - L'utilisateur peut imposer un séquencement des calculs. Le système va alors trouver les partitionnements, c'est-à-dire les distributions des données et des calculs en mémoire et sur les processeurs adéquats.
  - De même, l'utilisateur peut imposer un partitionnement initial, le système va trouver les ordonnancements compatibles.
- **de jouer sur les compromis (« trade-off ») usuels** : le procédé permet à l'utilisateur de valider les compromis entre

les paramètres sensibles du placement en phase de conception de l'application, compromis entre :

- Nombre de processeurs/Bande passante,
- Mémoire/temps réel,
- Nombre de processeurs/temps réel,
- Bande passante/temps réel,
- Mémoire/Bande passante.

- **de visualiser des solutions complexes** comme par exemple l'ordonnancement, le découpage des données, l'affectation etc...
- **de dimensionner la machine** : le procédé permet de spécifier les ressources nécessaires pour placer une application particulière sur un type de machine donnée sans violer les contraintes applicatives. Cela consiste à prendre en compte les dimensions et le nombre de chaque composant matériel (Hardware).
  - Nombre et puissance des processeurs,
  - Performance du réseau d'interconnexions,
  - Taille et type de mémoire (mémoire vive et mémoire cache synchrone/asynchrone, disque dur),
  - Dimension des différentes interfaces systèmes.
- **de choisir le critère d'optimisation du placement** :
  - **Dimension machine** : le procédé permet de configurer une machine minimale pour une application donnée. L'utilisateur peut, par exemple, choisir de configurer une machine avec un nombre minimal de processeurs.
  - **Latence** : le procédé permet de trouver le (ou les) placement(s) qui minimise(nt) le temps d'exécution de l'application sur une machine cible prédéfinie par l'utilisateur.
  - **Efficacité** : le procédé permet de maximiser le parallélisme du (ou des) placement(s) de l'application sur une machine cible prédéfinie par l'utilisateur.
  - **Coût** : en intégrant un coût (financier par exemple) sur chacun des composants matériels, le procédé permet de

trouver le (ou les) placement(s) de l'application qui minimise(nt) ce coût.

- **Temps d'occupation de la machine** : le procédé permet de trouver le (ou les) placement(s) qui minimise(nt) le temps d'occupation de la machine cible prédéfinie par l'utilisateur afin de pouvoir éventuellement placer une seconde application.
- **Récurrence d'entrée ou de sortie** : le procédé permet de trouver le (ou les) placement(s) qui minimise(nt) la cadence d'entrée et/ou de sortie des résultats produits par un ensemble de traitements. Ces contraintes sont souvent imposées dans le cadre des signaux vidéo, acoustiques et/ou hyperfréquences.

- **de déterminer les paramètres du signal** comme le modes des entrées/sorties, la latence, la récurrence (d'entrée et/ou de sortie) pour une machine et une application données.

De plus, la présence d'un ensemble d'heuristiques permet à l'utilisateur de soulager la recherche d'une solution, en l'orientant. L'ingénieur peut en effet ;

- **choisir une ou plusieurs heuristiques** dans un ensemble prédéfini, c'est-à-dire avoir le choix de différentes solutions canoniques de placement. Par exemple :
  - Ordonnancement des calculs au plus tôt, ou au plus tard.
- **Maximiser le parallélisme,**
- **Minimiser les communications,**
- **Recouvrir les communications par les calculs,**
- **Maximiser la localité des données dans les différents processeurs.**

## REVENDICATIONS

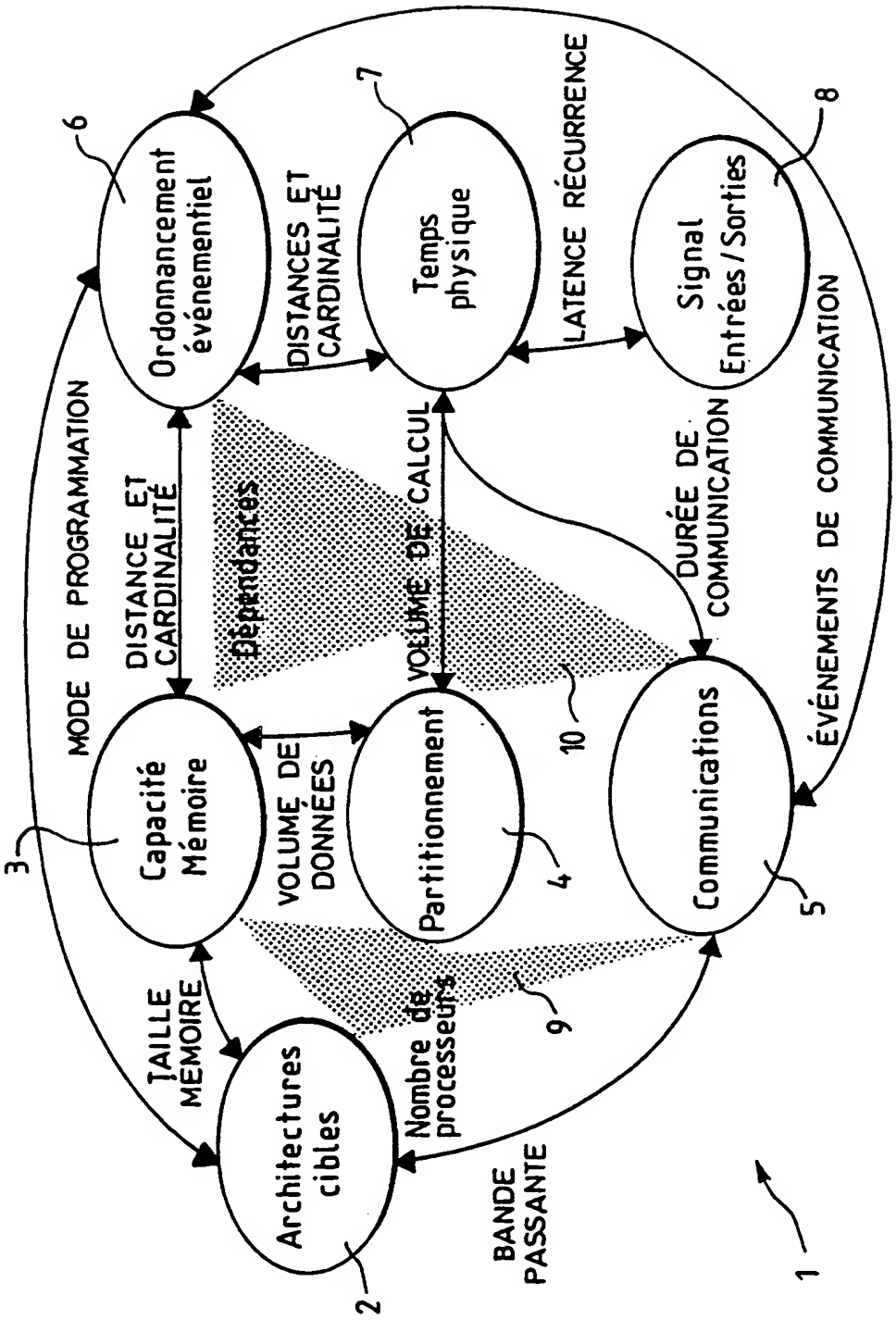
1. Procédé générique d'aide au placement d'une application de traitement de signal sur un calculateur à architecture parallèle homogène, caractérisé par le fait qu'il consiste, pour chaque constituant fonctionnel et physique de l'application, à établir un modèle défini par un ensemble de relations sur les différentes variables relatives à ce constituant, afin de modéliser les contraintes, à résoudre de façon concurrente les relations ainsi établies, et à en déduire au moins une solution, et à effectuer le placement de l'application.

2. Procédé selon la revendication 1, caractérisé par le fait que les différents modèles mis en œuvre sont : l'architecture des processeurs cibles (2), la capacité de la mémoire (3), le partitionnement des flots de données (4), les communications inter-processeurs (5), le séquencement des calculs (6), le temps de calcul (7) et les entrées-sorties de signaux (8).

3. Procédé selon la revendication 1 ou 2, caractérisé par le fait qu'en cas d'obtention de plusieurs solutions, on choisit celle optimisant au moins un critère, le choix étant fait de façon heuristique.

4. Procédé selon l'une des revendications précédentes, caractérisé par le fait que l'application est décrite fonctionnellement par un graphe de flot de données multidimensionnelles.

5. Procédé selon la revendication 4, caractérisé par le fait que l'application est décrite à partir de l'un des langages suivants : le langage « ARRAY-OL », le langage ALPHA, un langage décrivant un ensemble de nids de boucles parfaitement imbriqués, le formalisme de description de type MD/SDF.


Ordonnancement événementiel
Temps physique
Signal Entrées / Sorties
Capacité Mémoire
Partitionnement
Communications
Architectures cibles
DISTANCES ET CARDINALITÉ
LATENCE RÉCURRENCE
MODE DE PROGRAMMATION
DISTANCE ET CARDINALITÉ
VOLUME DE CALCUL
DURÉE DE COMMUNICATION
ÉVÉNEMENTS DE COMMUNICATION
VOLUME DE DONNÉES
TAILLE MÉMOIRE
Nombre de processeurs
BANDE PASSANTE
Dépendances
1
2
3
4
5
6
7
8
9
10